程式人 月刊雜誌

Programmer

捐發票愛心條碼

讀書做善事、寫書做公益 – 歡迎程式人認養專欄或捐出您的網誌

參考價： NT 50 元，如果您喜歡本雜誌，請將書款捐贈公益團體

羅慧夫顱顏基金會 彰化銀行 (009) 帳號： 5234-01-41778-800

程式人雜誌

2014 年 8 月號

本期焦點：FPGA 可程式化電路

程式人雜誌

前言

編輯小語

在本期的「程式人雜誌」中，聚焦的主題是「FPGA 技術與 Verilog 硬體描述語言」。

雖然「程式人」撰寫的通常是「軟體程式」，但是在 Maker 運動與 Prototyping 技術越來越發達的今天，程式人所能做的事情已經越來越多了。

我們可以用 Verilog 或 VHDL 設計「硬體描述程式」，然後轉為電路燒到 FPGA 板當中，也可以用電腦設計 3D 模型，然後用 3D 印表機列印出來，甚至可以用電腦設計電路，然後用像 Eagle 或 Altium Designer 這樣的軟體轉成 PCB 電路，之後再用雕刻機或洗電路的方式將電路做出來。

或許、在未來的某一天，我們可以把「3D 模型、Verilog、電路」等等都用電腦設計出來，然後用一台「超級印表機」，直接印出一台完整的「手機」了也說不定！

----（程式人雜誌編輯 - 陳鍾誠）

授權聲明

將會在該文章或素材後面標示其授權，此時該文章將以該標示的方式授權釋出，請修改者注意這些授權標示，以避免產生侵權糾紛。

例如有些文章可能不希望被作為「商業性使用」，此時就可能會採用創作共用：姓名標示、非商業性、相同方式分享 的授權，此時您就不應當將該文章用於商業用途上。

最後、懇請勿移除公益捐贈的相關描述，以便讓愛心得以持續散播！

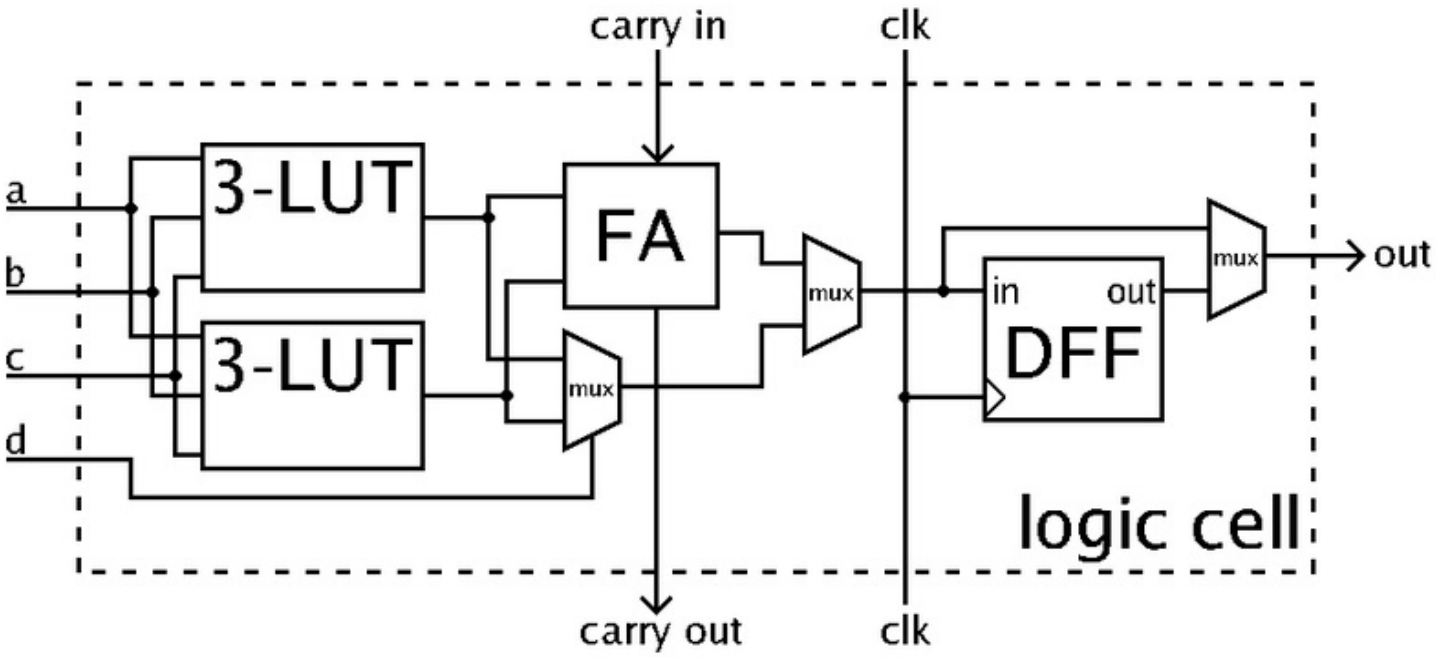
本期焦點

FPGA 簡介

在目前的數位電路技術當中，FPGA與CPLD是兩種可程式化的電路裝置，其中CPLD是永久性可程式化電路，只要燒錄進去之後就會永久存在，即使關機重開，電路仍然會存在的一種技術。而FPGA則是像 RAM一樣在關機後就會消失的可程式化電路，因此通常會搭配一塊 EPROM ，在開機的時候將電路燒錄到FPGA當中，讓電路看起來就像永久存在一樣。

圖、Altera StratixIVGX FPGA

FPGA 是由一種稱為 CLB (Configurable Logic Block) 或 LAB (Logic Array Block) 的基礎區塊所組成的。下圖是一個 CLB 區塊的典型結構，其中包含一個全加器 (FA)、一個 D-Type 的正反器、三個多工器 (mux) 與兩個三輸入的 Lookup tables (3-LUTs)。



圖、FPGA 的邏輯區塊 CLB

在上圖的 CLB 區塊中，如果走上面的全加器路徑，兩個 3-LUTs 的輸出會被相加，加上 carry-in 之後就成了一組完整的全加器電路。由於該全加器FA的輸入是3-LUTs 的輸出，因此 carry-out 的輸出結果可以寫成如下算式。

carry-out = FA(carry-in, 3-LUT(a,b,c), 3-LUT(a,b,c))

而且 3-LUT(a,b,c) 的輸出可以是一個三輸入的邏輯真值表，因此可以建構出任意的 a,b,c 等三輸入的邏輯電路組合。

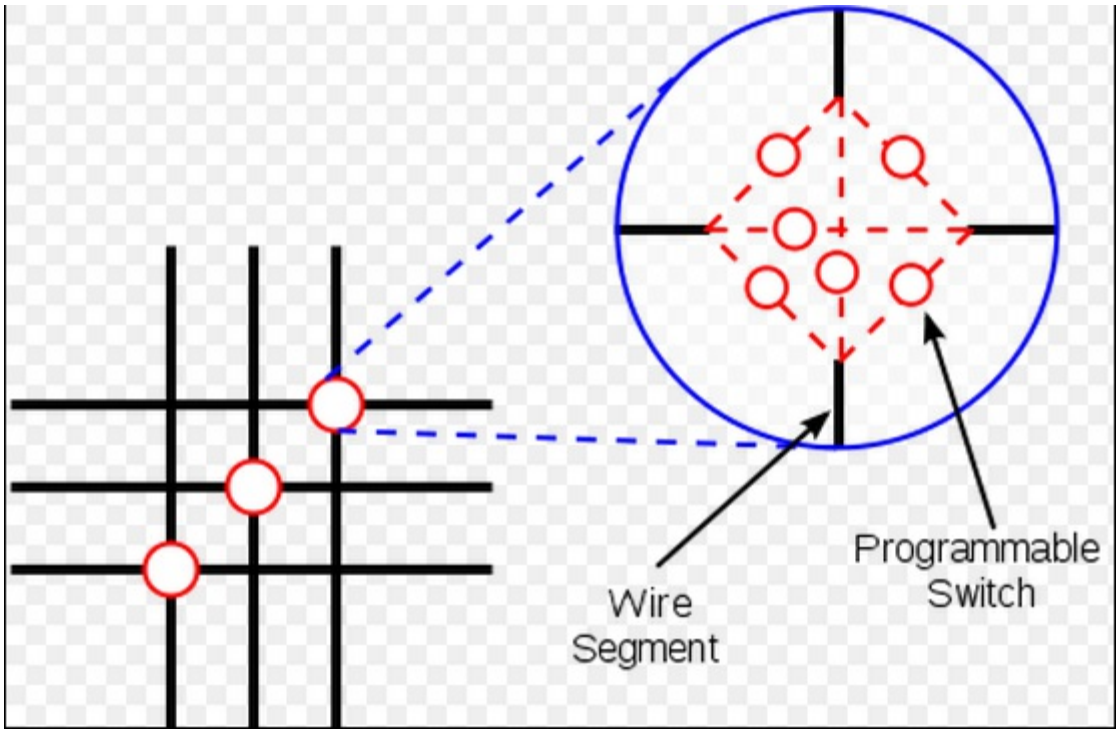
舉例而言，如果兩個 3-LUT 分別是 (a and b) or c 與 a xor (b and c) 的話，那輸出結果就會是

carry-out = FA(carry-in, (a and b) or c, a xor (b and c))

當然、如果走下面的 mux 路徑，就不會進行加法運算，而是進行二選一的多工運算，因此會在兩個LUT 的結果當中選擇一個作為輸出。

最後、這個輸出可能會經過 D 正反器 (DFF) 進行儲存動作後再輸出，或者是直接輸出到 out 線路中。

而這些 CLB 區塊會透過下列的 Switch Box 連接形成更大的區塊，每個 Switch Box 的結構如下圖所示。



圖、切換盒 Switch Box

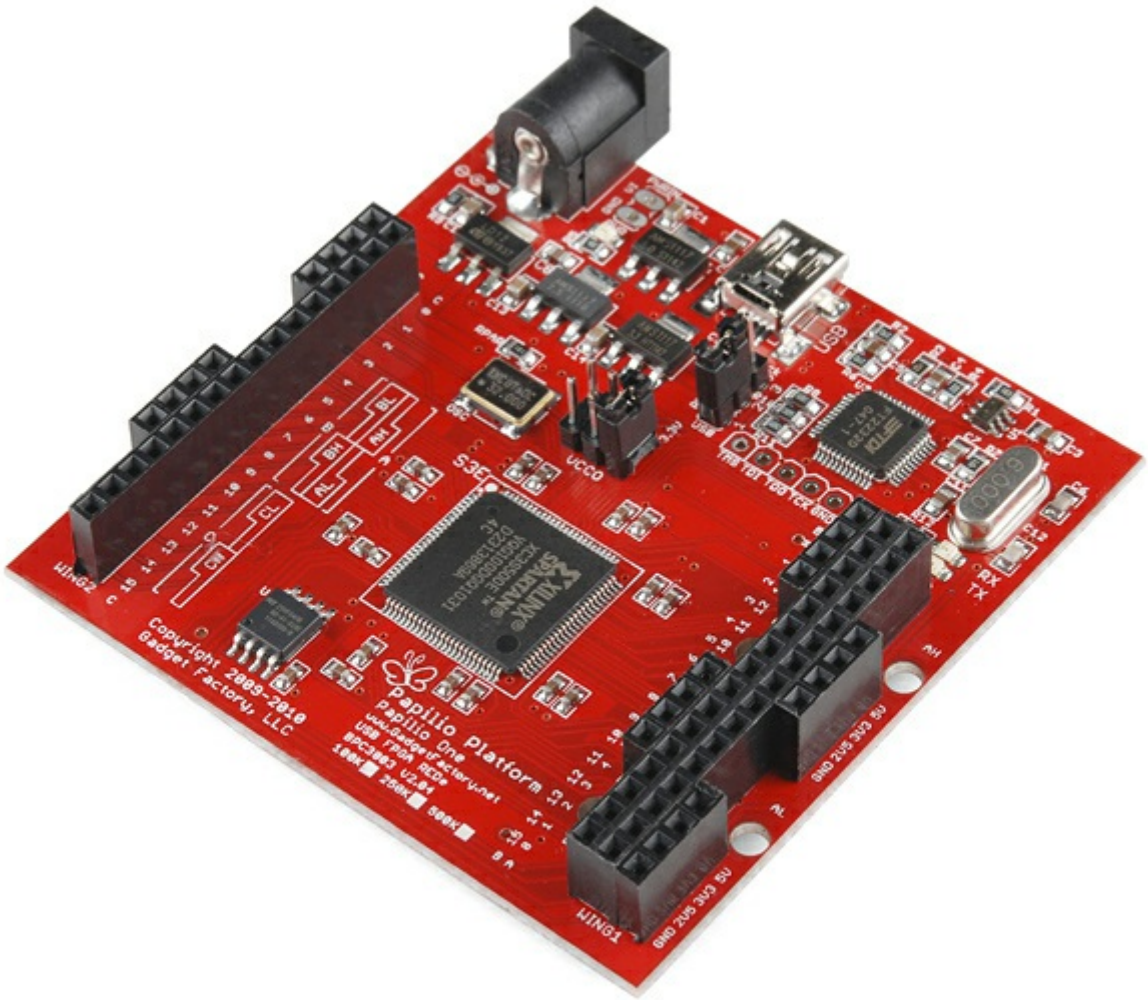
目前的 FPGA 晶片容量，通常包含數十萬到數百萬個 CLB 邏輯單元，因此已經可以用 FPGA 做出非常複雜的電路，甚至可以將上千顆處理器燒錄到這些 FPGA 晶片中，成為一台超級平行電腦。以下是 Xilinx 公司的 FPGA IC 規格表。

	Spartan-6	Artix-7	Kintex-7	Virtex-7	Kintex UltraScale	Virtex UltraScale
Logic Cells	147,443	215,360	477, 760	1,954,560	1,160,880	4,407,480
BlockRAM	4.8Mb	13Mb	34Mb	68Mb	76Mb	132.9Mb
DSP Slices	180	740	1,920	3,600	5,520	2,880
DSP Performance (symmetric FIR)	140GMACs	930GMACs	2,845GMACs	5,335GMACs	8,180 GMACs	4,268 GMACs
Transceiver Count	8	16	32	96	64	120
Transceiver Speed	3.2 Gb/s	6.6 Gb/s	12.5 Gb/s	28.05 Gb/s	16.3 Gb/s	32.75 Gb/s
Total Transceiver Bandwidth (full duplex)	50 Gb/s	211 Gb/s	800 Gb/s	2,784 Gb/s	2,086 Gb/s	5,886 Gb/s
Memory Interface (DDR3)	800	1,066	1,866	1,866	2,400	2,400
PCI Express® Interface	x1 Gen1	x4 Gen2	x8 Gen2	x8 Gen3	x8 Gen3	x8 Gen3
Analog Mixed Signal (AMS)/XADC	-	XADC	XADC	XADC	System Monitor	System Monitor
Configuration AES	Yes	Yes	Yes	Yes	Yes	Yes
I/O Pins	576	500	500	1,200	832	1,456
I/O Voltage	1.2V – 3.3V	1.2V – 3.3V	1.2V – 3.3V	1.2V – 3.3V	1.0 – 3.3V	1.0 – 3.3V
Please refer to the device data sheets for the latest product information.						

圖、 Xilinx 的各款 FPGA IC 容量規格表(以上表格來自 Xilinx 的網站，網址為： http://www.xilinx.com/products/silicon-devices/fpga/)

目前FPGA的兩大廠商是 Xilinx 與 Altera，這兩家廠商供應 FPGA IC、開發板與設計工具軟體等產品，而其他的小廠則通常採購這兩家的 IC 放入自己設計的電路板中，因為 IC 的設計與生產成本通常較高，所以小廠通常沒有足夠的經費去設計與量產 FPGA IC。

在開放原始碼的領域，目前我們看到有 Gadget Factory 這家公司釋出了開放的 FPGA 電路板，該公司已經生產了 Papilio One, Papilio Pro 與 Papilio DUO 等三款 FPGA 電路板。 Papilio 電路板的特色是具有模仿 arduino 開發板的功能，並且修改了 arduino 的開發工具，讓您可以在 Papilio FPGA 板中撰寫 arduino 程式。以下是 Papilio One 電路板的圖片 。



圖、Papilio One FPGA 電路板

在台灣，北瀚科技 (SMIMS) 也有設計生產 FPGA 電路板，筆者目前正與北瀚科技洽談有關設計開放原始碼的 FPGA 軟硬體當中，希望可以透過這個合作創建一台從軟體、硬體到 CPU 全部開放的電腦。當然、這將會是一台 FPGA 電腦，這樣的設計雖然效能不見得很好，但是卻會是一台非常適合學校教學研究用的電腦。

筆者已經為這個目標撰寫了一本書，連結如下：

- 開放電腦計畫-計算機硬體結構 (使用 Verilog 實作)

如果與北瀚科技的合作能成功的話，我們會將「軟體、硬體與書籍」全部都以開放原始碼的方式公開發行。

目前北瀚科技已經釋出了兩塊開放原始碼的電路板在 github 上，網址如下：

- https://github.com/ccckmit/OpenFpgaBoard

但是其中有一顆 SMIMS 引擎與對應的程式還沒開放原始碼，所以筆者還在尋求完整開放的可能性，讓台灣的公司也能以更開放的態度面對 open source 社群。

後記：筆者認為、過去幾十年台灣有很好的電腦硬體製造基礎，因此有很多開放硬體的人才，他們有能力設計出像 arduino 或樹莓派 (raspberry Pi) 這樣的硬體與軟體，但是由於缺乏開放原始碼的思維，因此 arduino 與樹莓派都不是台灣的公司或組織所設計出來的。筆者相信、在當今的世界中，開放的心態會是很重要的，透過開放硬體的方式，或許可以為日漸衰落的台灣硬體產業，找到一條全新的道路也說不定。

FPGA 的設計流程與開發工具 -- 使用 Icarus

+ Altera Quartus II + 北瀚 FPGA 板子

FPGA 是一種可程式化硬體， 所以我們可以「寫程式」燒到FPGA裡面，但是要怎麼寫，又該怎麼燒呢？

撰寫 FPGA 程式通常要採用「硬體描述語言」 (Hardware Description Language, HDL)，目前最常被使用的「硬體描述語言」有兩種，一種是 Verilog，另一種是 VHDL。VHDL 在學術界很常用，很多學校課堂上教授「數位電路設計」或「數位系統設計」的時候都會採用 VHDL；而 Verilog 則是在業界比較常用，而且語法相對精簡，比較不需要重複宣告，因此寫起來很輕鬆愉快，這也是筆者為何喜歡 Verilog 的原因。

當我們寫好一個 Verilog 或 VHDL 程式模組的時候，通常會寫一段稱為 testbench 的測試程式，來測試該模組是否能正常運作，這個過程完全可以在電腦上執行，電腦會根據 Verilog 語法模擬電路的執行過程，不需要立刻燒錄到 FPGA 當中。

舉例而言，開放原始碼的 icarus 是筆者很喜歡使用的 Verilog 模擬測試工具，以下我們就先用 icarus 這個測試工具來說明 verilog 程式的設計與測試流程。

首先，讓我們先撰寫一個超級簡單的模組，並命名為 simple.v 後存檔：

檔案： simple.v

```
module simple(input clock, input i, output o);
  assign o = i;
endmodule
```

上述模組在硬體上其實只是一條線，這條線從 i 點的輸入拉到 o 點的輸出 (雖然還有一個 clock 的參數，但是在此處並未用到，宣告 clock 的原因單純是因為我們接下來要使用的 SMIMS Instrument 工具要求模組必須有時脈 clock 才能連接，所以就多宣告了這個參數，預留給後面綁定時使用的)。

接著、我們就可以另外寫一段測試主程式 (test bench)，並且命名為 simpleTest.v 後存檔:

檔案： simpleTest.v

```
`include "simple.v"

module main;
reg clock, i;
wire o;

simple s1(clock, i, o);

initial begin
  $monitor("%4dns i=%b o=%b", $stime, i, o);
  clock = 0;
  i = 0;
  #30 i=1;
  #100 i=0;
  #100 i=1;
  #100 $finish;
end

always #10 clock=~clock;

endmodule
```

接著我們就可以用 icarus 的 iverilog 指令對 simpleTest.v 這個檔案進行編譯動作，接著再用 vvp 這個指令執行測試程式，過程如下所示：

```
D:\SMIMS\ccc\simple>iverilog simpleTest.v -o simpl
eTest

D:\SMIMS\ccc\simple>vvp simpleTest
   0ns i=0 o=0
  30ns i=1 o=1
 130ns i=0 o=0
 230ns i=1 o=1
```

當測試完成，模擬結果沒有問題之後，就可以進一步將程式模組真正燒錄到 FPGA 板上去了，在此我們採用北瀚科技 (SMIMS) 的「VeriLite Altera C4」FPGA 板作為範例。

圖、北瀚科技 (SMIMS) 的 VeriLite Altera C4 FPGA 板

由於北瀚科技是小廠，不像 Altera 與 Xilinx 等大廠有自己的 IC 與完整的

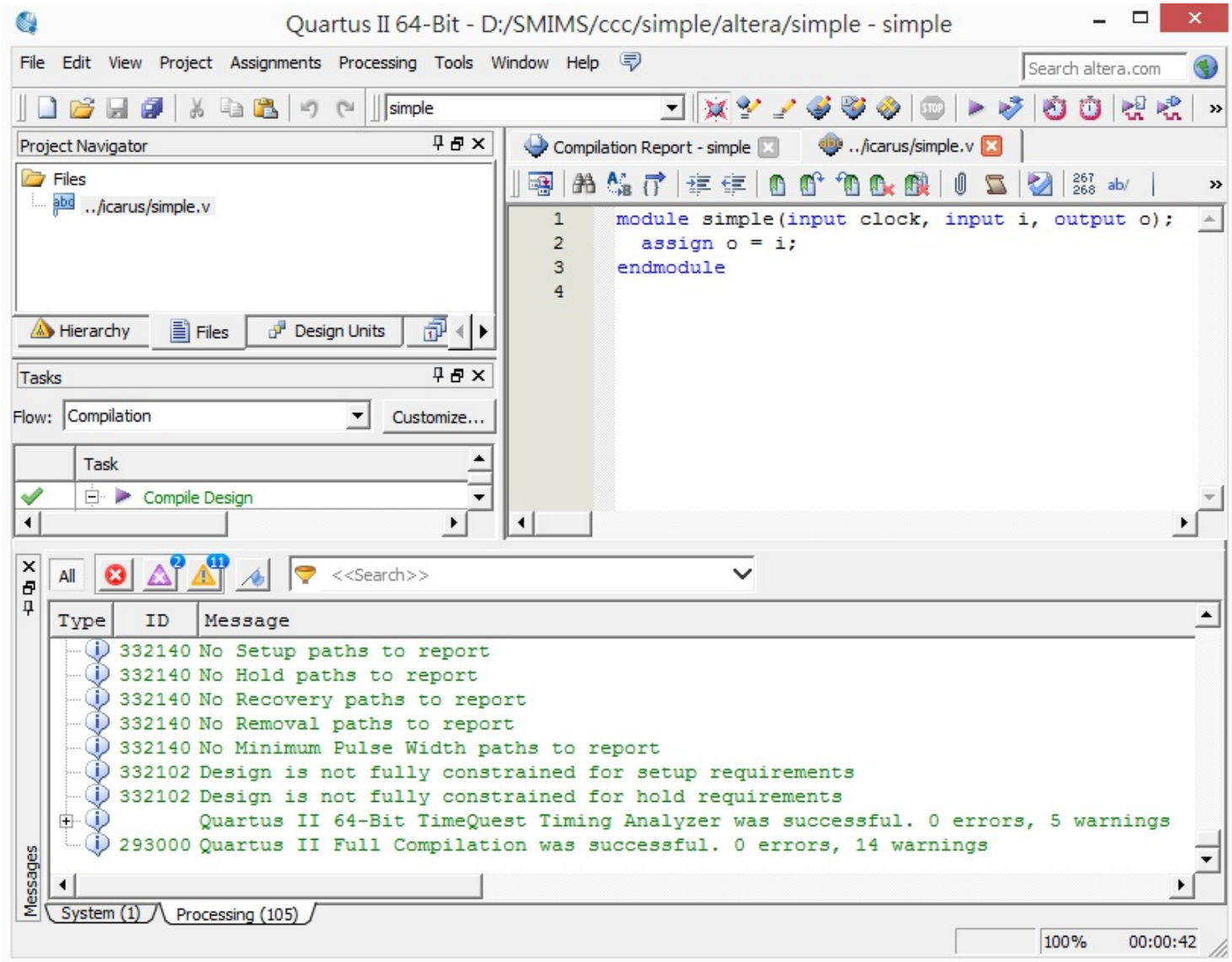
開發工具，因此北瀚的工具必須搭配上述兩大廠的開發工具使用。

目前、Xilinx 與 Altera 是 FPGA 的兩大廠商，這兩家廠商都提供了FPGA的設計開發工具，Altera 的開發工具稱為 Quartus II，Xilinx 的開發工具稱為 ISE，這兩者都提供了相當完整的工具鏈，包含 Verilog、VHDL、測試檔撰寫、模擬、燒錄、腳位設定等工具。

另外、Xilinx 的 ISE 與 Altera 的 Quartus II 都提供了用圖形化設計模組的方式，設計完之後也可以產生 Verilog 與 VHDL 的程式碼，這種工具有助於學習，但是對於專家而言，直接撰寫程式會是更快速有效的方法。

筆者並沒有購買 Altera Quartus II 軟體，因此使用的是免費的 Quartus II web edition，但是對於初學者而言，這個版本已經是相當好用的了。

以我們上述的 VeriLite Altera C4 FPGA 板為例，我們必須先用 Altera 的 Quartus II 工具來編譯並產生 FPGA 燒錄檔，因此我們必須用 Quartus II 建立一個專案，並在建立時指定使用的 FPGA 晶片型號，相關模組等資訊，然後才能進行編譯，以下是筆者用 Quartus II 編譯 simple.v 完成時的畫面。

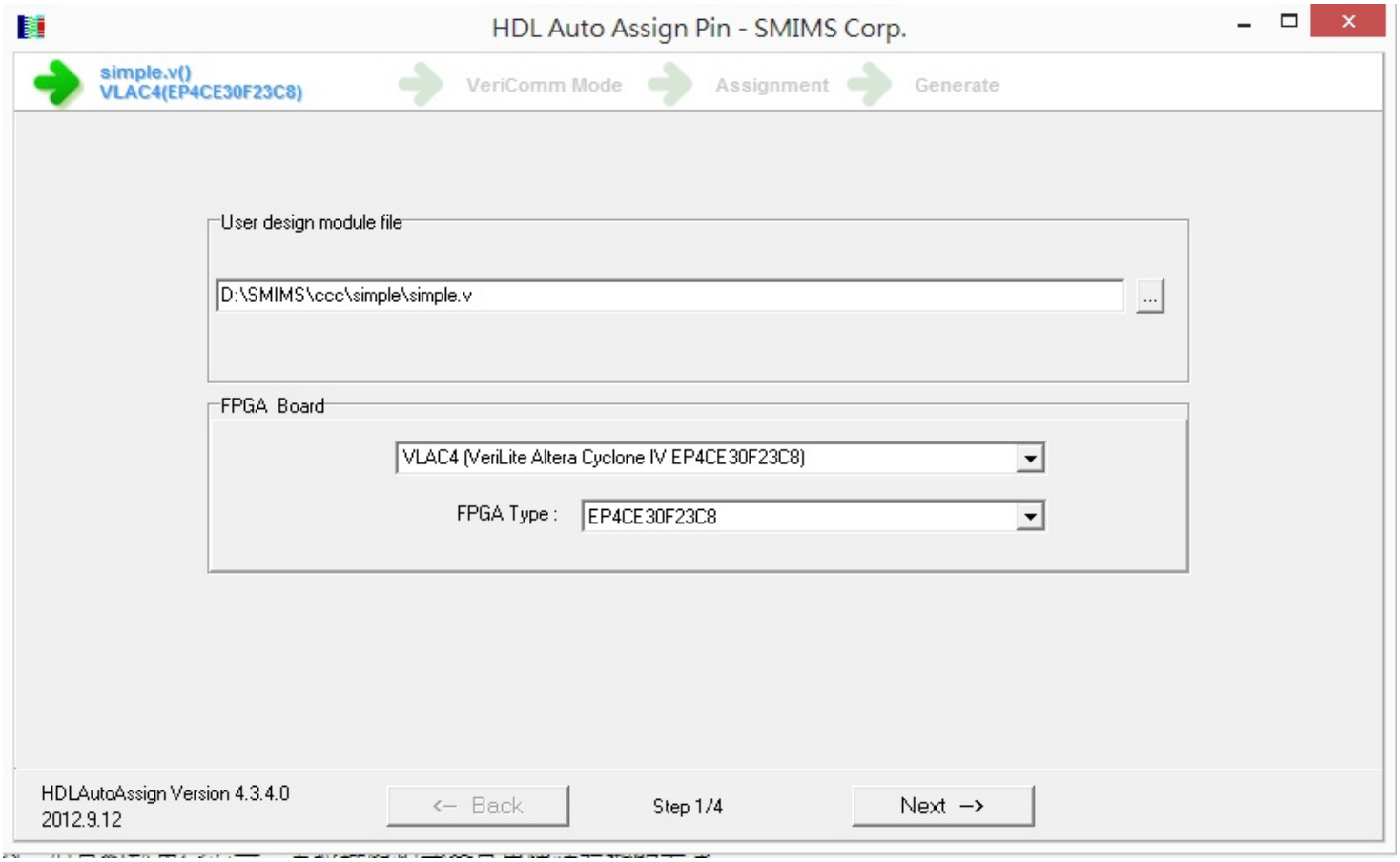


圖、使用 Altera Quartus II 編譯 simple.v 模組

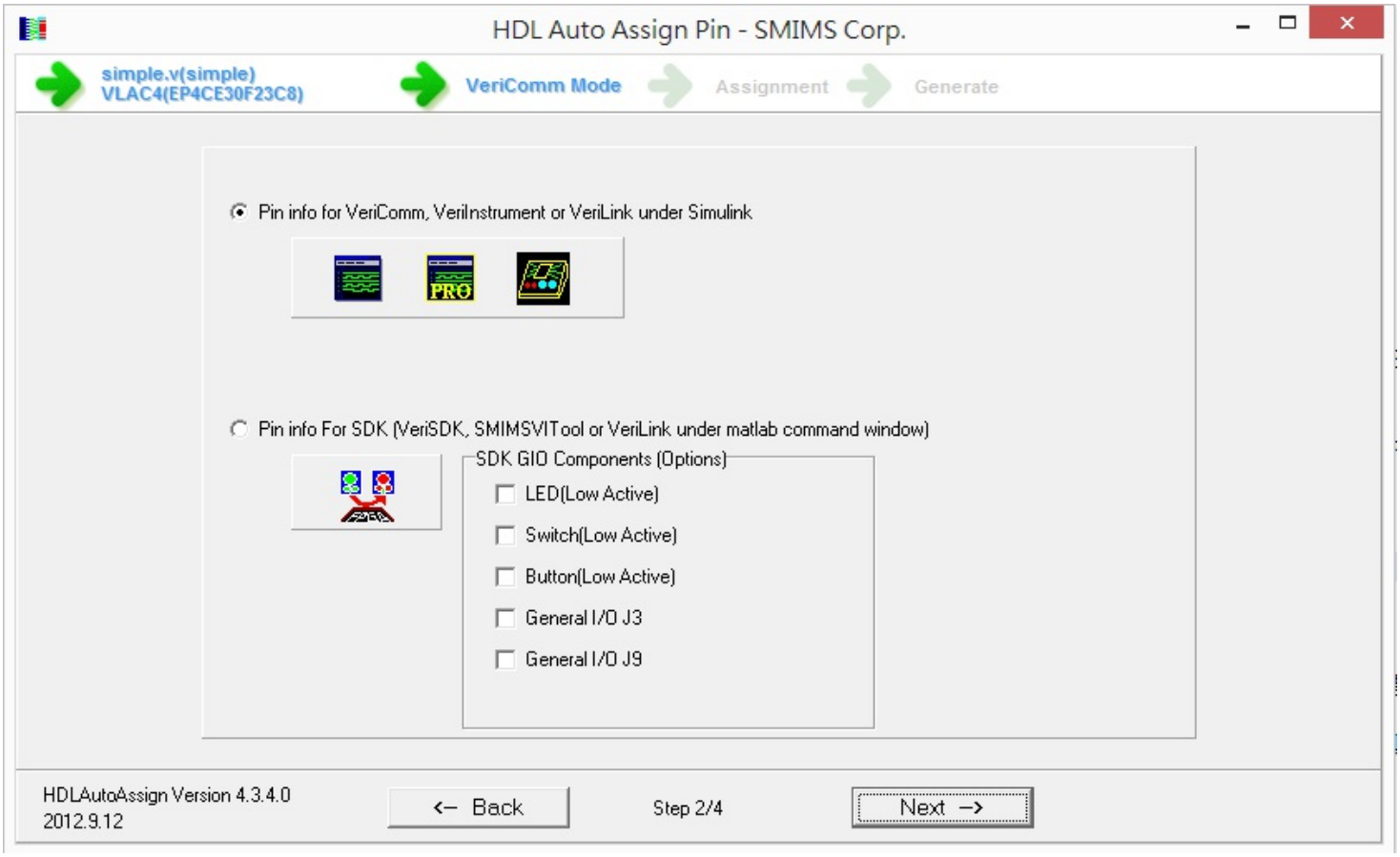
但是、FPGA 的「程式設計」流程，與一般程式的設計流程有所不同。在寫一般程式的時候，我們會進行「撰寫、執行」或者「撰寫、編譯、執行」的流程，但是在寫 FPGA 程式的時候，還得加上「腳位綁定」與「燒錄」這兩個步驟。

「腳位綁定」是將 Verilog 模組的輸出入線路，與 FPGA 上的針腳對映起來的一個動作，綁定好之後必須重新進行 FPGA 等級的完整編譯，完成之後才能進行燒錄。

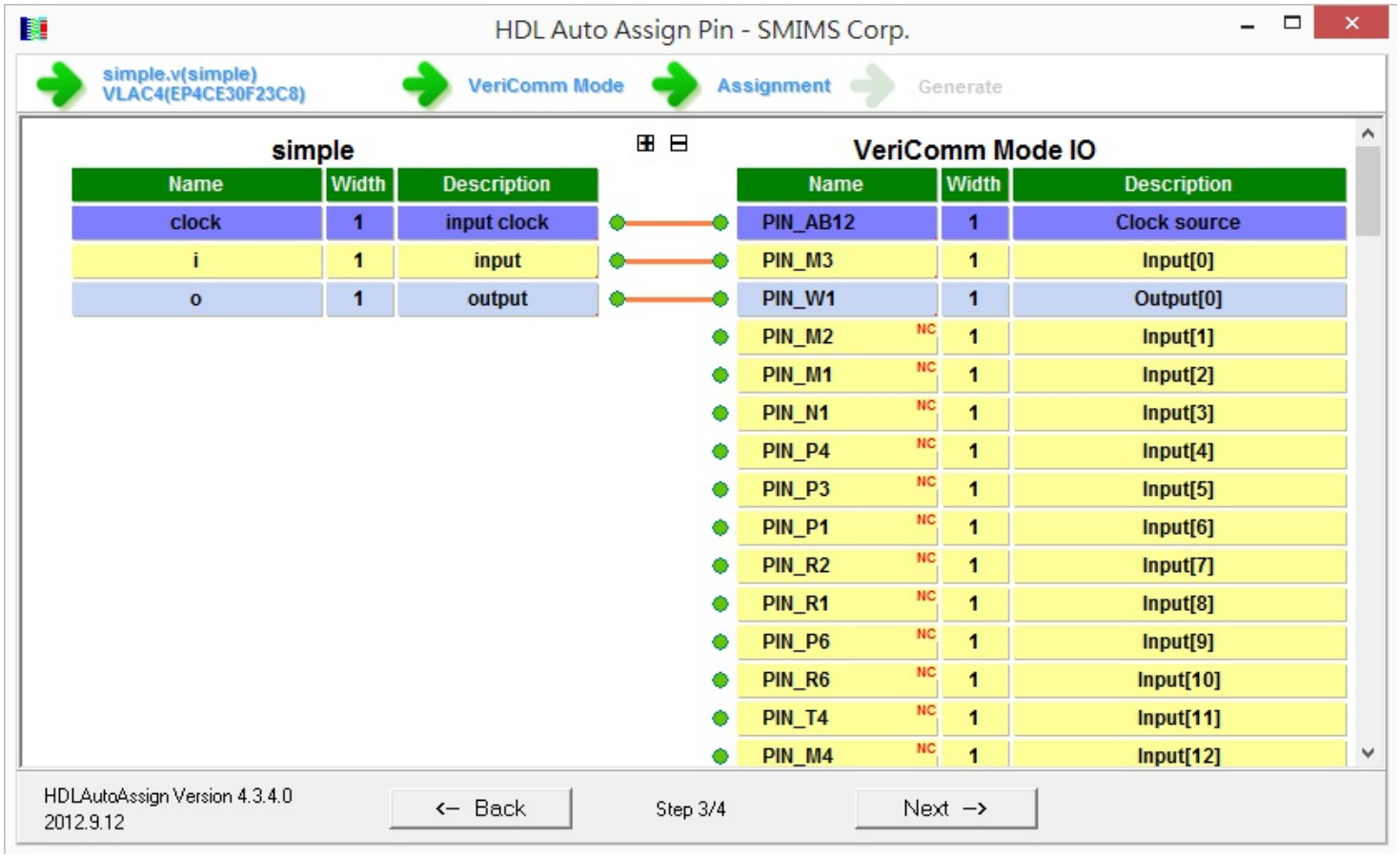
在 北瀚所提供的工具中，HDL Auto Assign Pin 這個工具來進行腳位的自動指定，其操作如下圖所示：



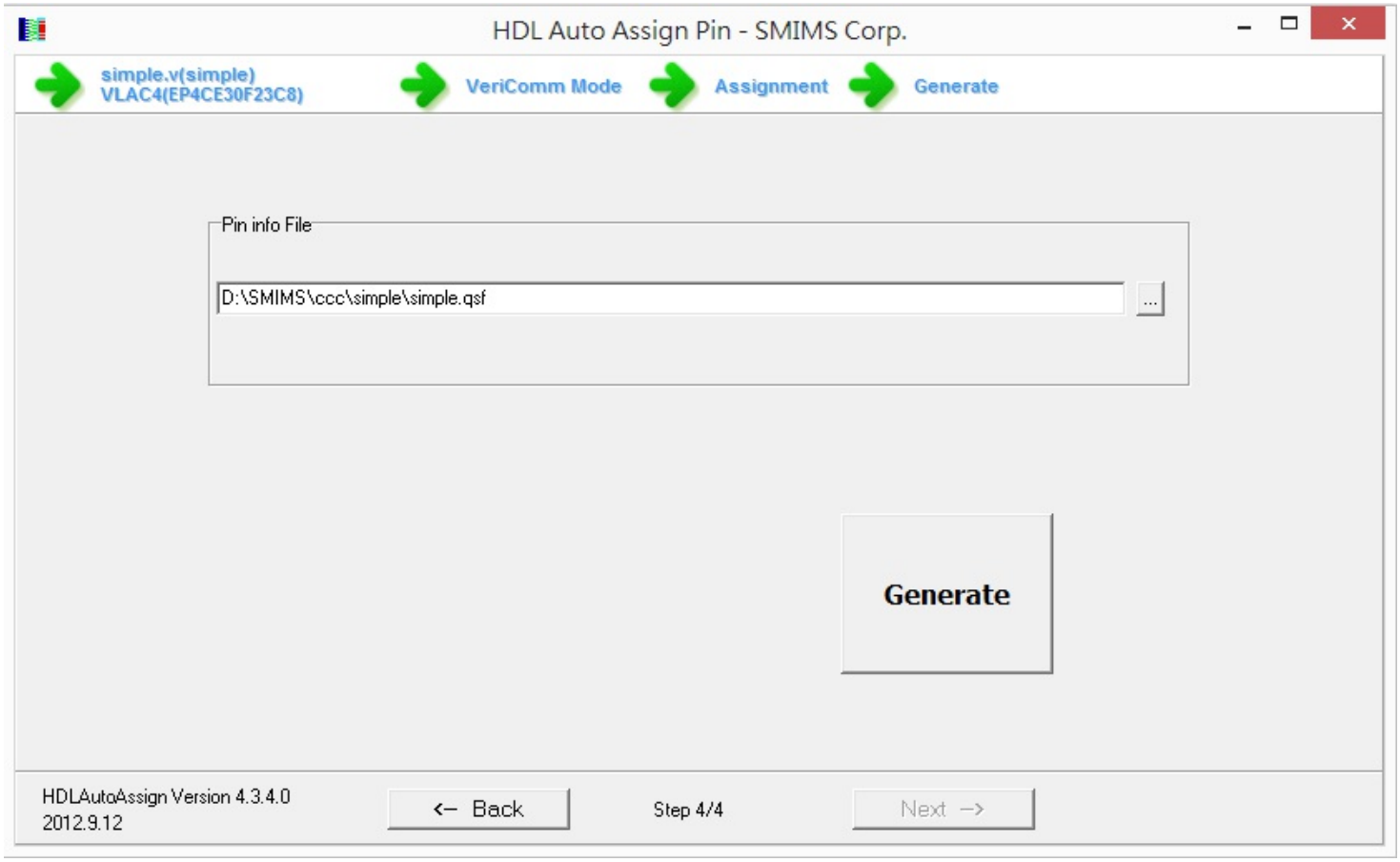
圖、北瀚的腳位自動綁定工具 - 步驟1



圖、北瀚的腳位自動綁定工具 - 步驟2



圖、北瀚的腳位自動綁定工具 - 步驟3

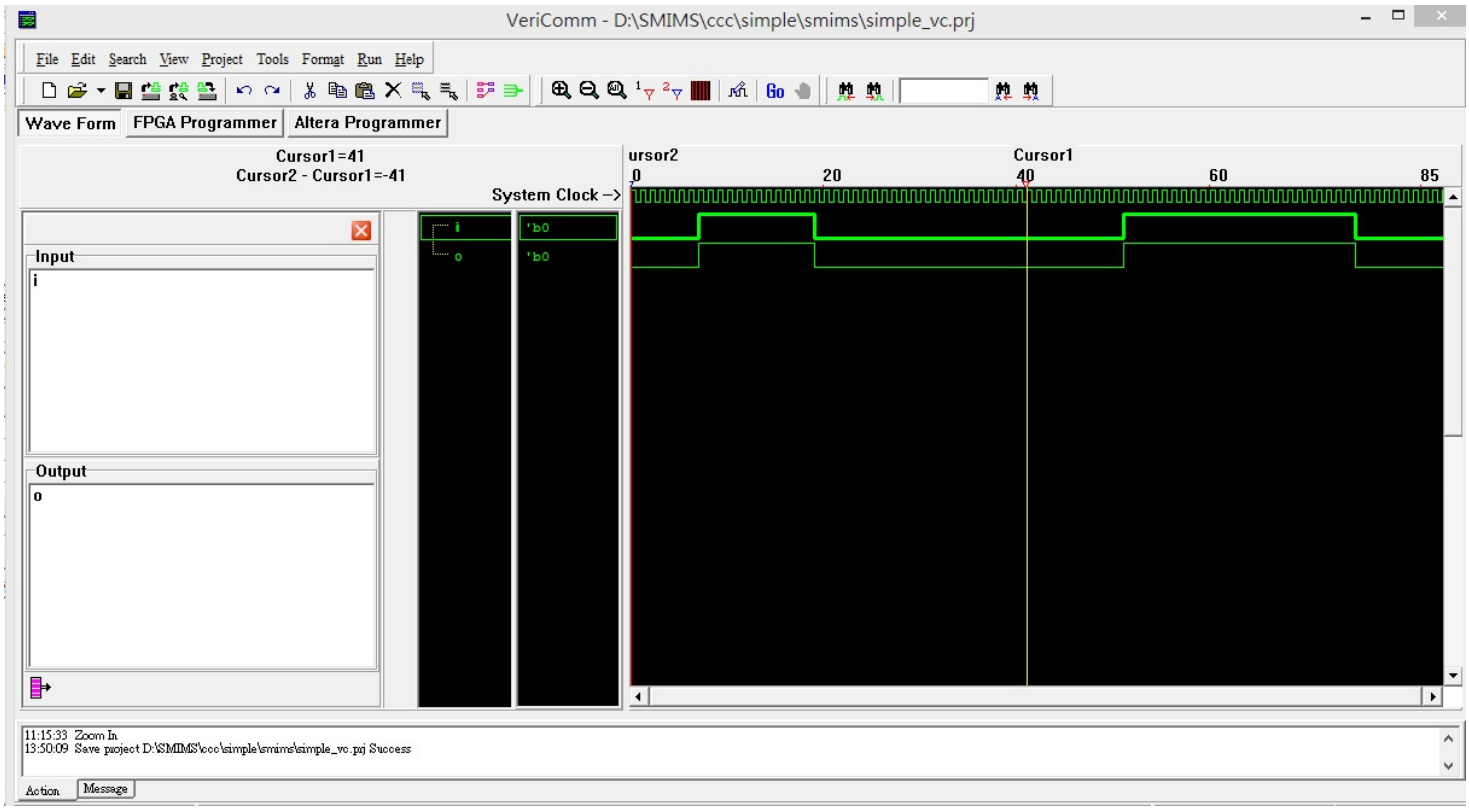


圖、北瀚的腳位自動綁定工具 - 步驟4

如果我們在編譯時沒有指定腳位資訊，Altera Quartus II 編譯出來就不會包含燒錄檔 (*.rbf)，因此我們可以用上述北瀚 HDL Auto Assign Pin 產生的檔案 simple.qsf 將 Quartus 編譯時預設產生的 simple.qsf 檔覆蓋過去，然後重新編譯一次，這樣才會正確的指定腳位，也才能產生正確的燒錄檔 simple.rbf。

(注意：最好將 北瀚 (SMIMS) 的檔案與 quartus 的檔案分開存放，否則可能會有互相覆蓋的問題)

當燒錄檔 simple.rbf 檔產生之後，我們就可以透過北瀚的 VeriComm 這個工具程式，將 simple.rbf 燒到北瀚的 FPGA 板中，然後編輯輸入波形，並且繪製出實際在 FPGA 板上跑出來的波形圖，如下所示：

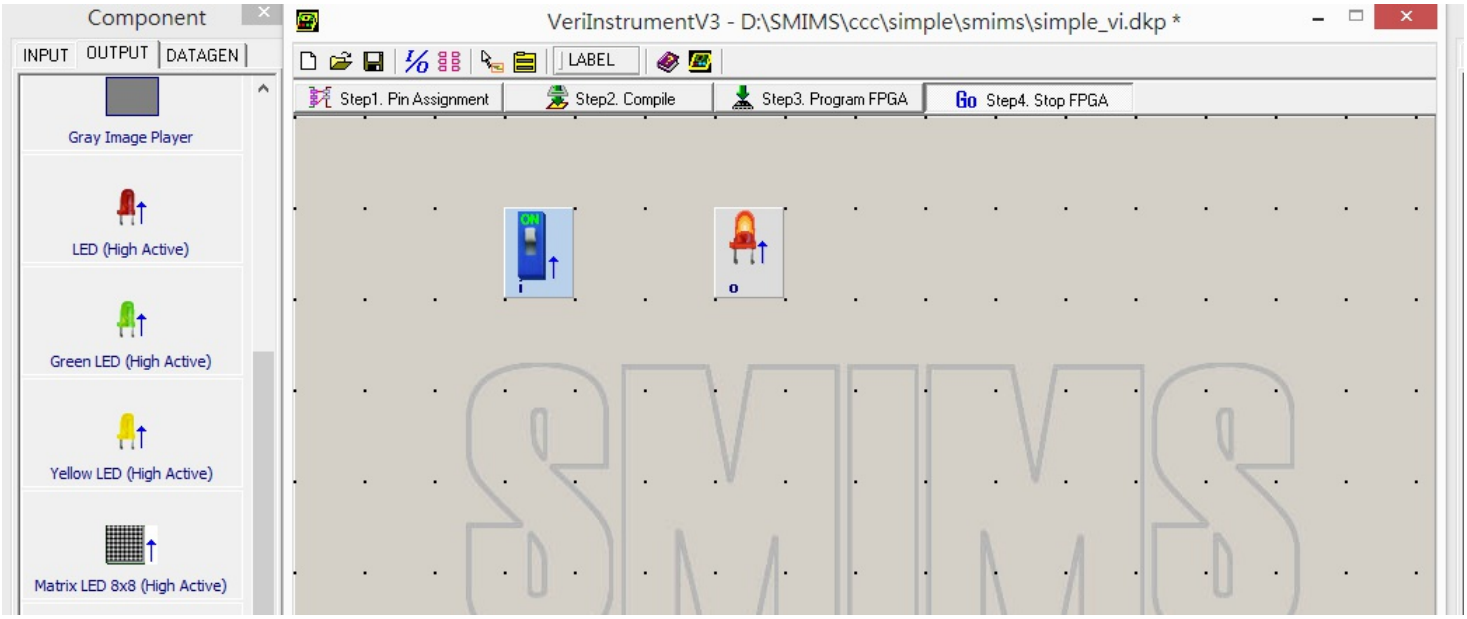


圖、VeriComm 程式執行出來的波形圖

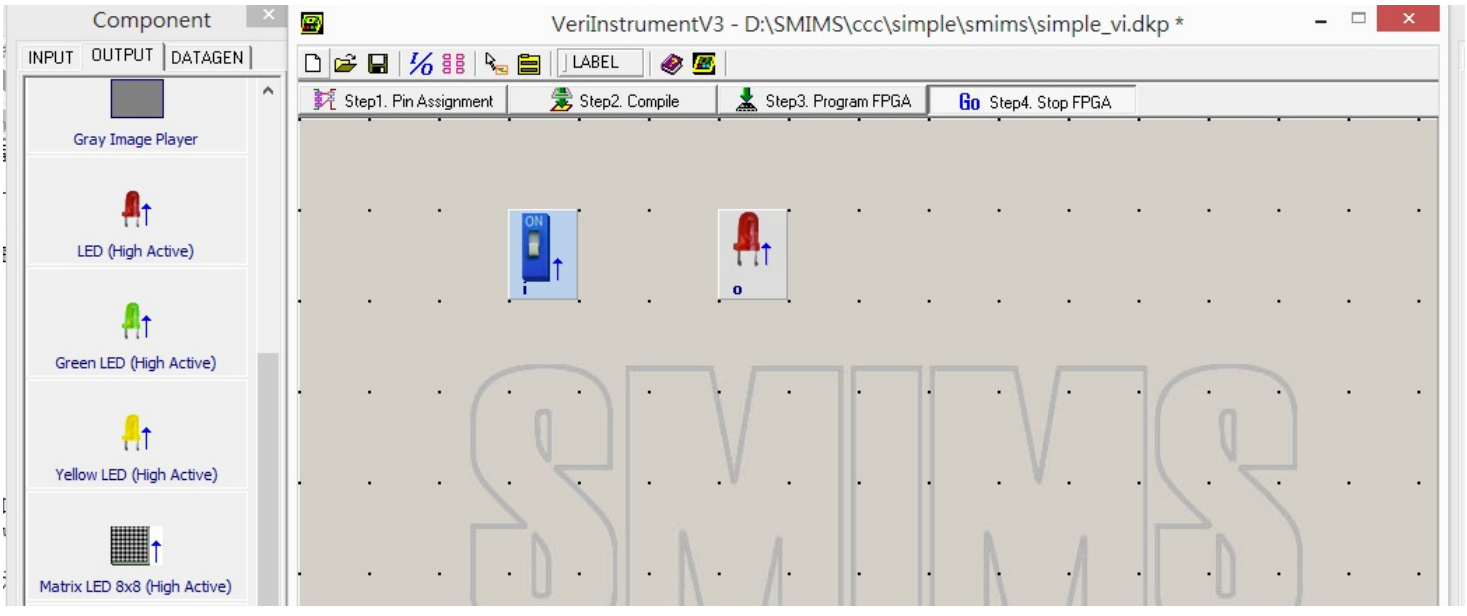
我們可以透過 VeriComm 輸出的這個波形看看所設計的 Verilog 程式是否正確，這個波形已經不是模擬的結果了，而是 VeriComm 透過控制 simple.v 的輸入與時脈，實際在 FPGA 板上對每個腳位進行偵測所繪製出來的，也就是讓 simple.v 在 VeriComm 與 SMIMS Engine 晶片控制下的輸出結果。

透過 VeriComm 的波形，我們會比較容易觀察所設計的電路是否正確，這也是北瀚科技比較特別的技術之所在。

接著、我們還可以利用北瀚的 VeriInstrument 工具，利用視覺化的方式，做出一個互動式的「虛擬電路」。舉例而言，以下是我將 simple.v 連結到「按鈕與LED」等圖示控制項，所得到的幾個畫面。



圖、VeriComm 程式執行出來的波形圖 - 開關向上扳(通路)



圖、VeriComm 程式執行出來的波形圖 - 開關向下扳(斷路)

至此、我們已經用一個極度簡單的範例，大致介紹完整個 FPGA 電路的設計過程，希望能讓大家對 FPGA 的電路設計有完整的概念。當然、學習

FPGA 的過程有些是很難以文字表達的，因此筆者錄了以下這些影片，希望能幫助讀者更生動的理解整個 FPGA 的電路設計的動態過程。

- YouTube 影片：FPGA 電路設計流程:用北瀚的板子示範 1 (整體流程介紹)
- YouTube 影片：FPGA 電路設計流程:用北瀚的板子示範 2 (icarus)
- YouTube 影片：FPGA 電路設計流程:用北瀚的板子示範 3 (quartus)
- YouTube 影片：FPGA 電路設計流程:用北瀚的板子示範 4 (pin assign)
- YouTube 影片：FPGA 電路設計流程:用北瀚的板子示範 5 (veriComm 波形偵測驗證)
- YouTube 影片：FPGA 電路設計流程:用北瀚的板子示範 6 (用 veriInstrument 接上虛擬周邊)

開放電腦計畫（13）-- 將 MCU0 放上 FPGA 執行（作者：陳鍾誠）

前言

先前我們曾經用流程式撰寫法設計過 MCU0 的迷你版與完整版，也曾經用區塊式的方法設計過 MCU0 的迷你版，這些文章列表如下：

- 開放電腦計畫 (6) – 一顆只有 51 行 Verilog 程式碼的 16 位元處理器 MCU0
- 開放電腦計畫 (7) – 完整指令集的 16 位元處理器 MCU0s
- 開放電腦計畫 (12) – 使用區塊式方法設計 MCU0 的 Verilog 程式

但是在這些文章中，我們都只用 icarus 去跑模擬測試，並沒有真正將 MCU0 放到 FPGA 上面跑，所以我們將在本文當中用北瀚的 FPGA 板來運行 MCU0 迷你版，讓 MCU0 真正成為「硬體」。

修改程式並模擬測試

為了讓 MCU0 能上 FPGA 跑， 我們必須修改一些程式，主要是因為 Verilog 在模擬的時候可以跑 initial 區段，但上 FPGA 時就沒有辦法執行 initial 區段，也不能用 readmemh() 讀取機器碼的十六進位檔了。

因此、我們只好加上重置訊號 reset ，並將機器碼在 reset 時直接塞入記憶體中，於是整個程式改寫如下：

檔案：muc0m.v

```
`define N     SW[15] // 負號旗標
`define Z     SW[14] // 零旗標
```

```
`define OP   IR[15:12] // 運算碼
`define C    IR[11:0]  // 常數欄位
`define M    {m[`C], m[`C+1]}

module mcuOm(input reset, input clock, output reg
 [15:0] A, output reg [15:0] IR, output reg [15:0]
 PC);
  parameter [3:0] LD=4'h0,ADD=4'h1,JMP=4'h2,ST=4'h
3,CMP=4'h4,JEQ=4'h5;
  reg [15:0] SW;  // 狀態暫存器
  reg [15:0] pc0;
  reg [7:0]  m [0:32];   // 內部的記憶體

  always @(posedge clock or posedge reset) begin

    if (reset) begin
      PC = 0;
      SW = 0;
      {m[0],m[1]}   = 16'h0016;  // 00    LOOP:
LD    I
      {m[2],m[3]}   = 16'h401A;  // 02
CMP   K10
      {m[4],m[5]}   = 16'h5012;  // 04
JEQ   EXIT
      {m[6],m[7]}   = 16'h1018;  // 06
```

```
ADD   K1
      {m[8],m[9]}    = 16'h3016;  // 08
ST    I
      {m[10],m[11]} = 16'h0014;  // 0A
LD    SUM
      {m[12],m[13]} = 16'h1016;  // 0C
ADD   I
      {m[14],m[15]} = 16'h3014;  // 0E
ST    SUM
      {m[16],m[17]} = 16'h2000;  // 10
JMP   LOOP
      {m[18],m[19]} = 16'h2012;  // 12    EXIT:
JMP   EXIT
      {m[20],m[21]} = 16'h0000;  // 14    SUM:
WORD  0
      {m[22],m[23]} = 16'h0000;  // 16    I:
WORD  0
      {m[24],m[25]} = 16'h0001;  // 18    K1:
WORD  1
      {m[26],m[27]} = 16'h000A;  // 1A    K10:
WORD  10
    end else begin
      IR = {m[PC], m[PC+1]};  // 指令擷取階段: IR=
m[PC], 2 個 Byte 的記憶體
      pc0= PC;                // 儲存舊的 PC 值在
```

```
pc0 中。
        PC = PC+2;                  // 擷取完成，PC 前進
到下一個指令位址
        case (`OP)                  // 解碼、根據 OP 執
行動作
          LD: A = `M;               // LD C
          ST: `M = A;               // ST C
          CMP: begin `N=(A < `M); `Z=(A==`M); end /
/ CMP C
          ADD: A = A + `M;          // ADD C
          JMP: PC = `C;             // JMP C
          JEQ: if (`Z) PC=`C;       // JEQ C
        endcase
        // 印出 PC, IR, SW, A 等暫存器值以供觀察
        $display("%4dns PC=%x IR=%x, SW=%x, A=%d", $
stime, pc0, IR, SW, A);
      end
  end
endmodule
```

先前我們通常將「模組與測試程式」寫在一起，但是為了放上 FPGA，我們決定把「模組與測試程式」分開，然後在測試程式當中用 include 的方式引用模組。

檔案：mcu0mTest.v

```
`include "mcu0m.v"
```

```
module main;                    // 測試程式開始
reg clock, reset;
wire [15:0] A, IR, PC, SW;
mcu0 mcu(reset, clock, A, IR, PC);   // 宣告 cpu0mc 處理器

initial begin
  clock = 0;            // 一開始 clock 設定為 0
  #10;
  reset = 1;
  #30;
  reset = 0;
end
always #10 clock=~clock;    // 每隔 10ns 反相，時脈週期為 20ns
initial #2000 $finish;      // 在 2000 奈秒的時候停止測試。
endmodule
```

接著、讓我們用 icarus 測試看看這個程式的運作是否正常，測試過程如下：

```
D:\SMIMS\ccc\mcu0m\icarus>iverilog mcu0mTest.v -o mcu0mTest
```

```
D:\SMIMS\ccc\mcu0m\icarus>vvp mcu0mTest
  50ns PC=0000 IR=0016, SW=0000, A=    0
  70ns PC=0002 IR=401a, SW=8000, A=    0
  90ns PC=0004 IR=5012, SW=8000, A=    0
 110ns PC=0006 IR=1018, SW=8000, A=    1
 130ns PC=0008 IR=3016, SW=8000, A=    1
 150ns PC=000a IR=0014, SW=8000, A=    0
 170ns PC=000c IR=1016, SW=8000, A=    1
 190ns PC=000e IR=3014, SW=8000, A=    1
 210ns PC=0010 IR=2000, SW=8000, A=    1
 230ns PC=0000 IR=0016, SW=8000, A=    1
 250ns PC=0002 IR=401a, SW=8000, A=    1
 270ns PC=0004 IR=5012, SW=8000, A=    1
 290ns PC=0006 IR=1018, SW=8000, A=    2
 310ns PC=0008 IR=3016, SW=8000, A=    2
 330ns PC=000a IR=0014, SW=8000, A=    1
 350ns PC=000c IR=1016, SW=8000, A=    3
 370ns PC=000e IR=3014, SW=8000, A=    3
 390ns PC=0010 IR=2000, SW=8000, A=    3
 410ns PC=0000 IR=0016, SW=8000, A=    2
 430ns PC=0002 IR=401a, SW=8000, A=    2
 450ns PC=0004 IR=5012, SW=8000, A=    2
 470ns PC=0006 IR=1018, SW=8000, A=    3
 490ns PC=0008 IR=3016, SW=8000, A=    3
 510ns PC=000a IR=0014, SW=8000, A=    3
```

```
 530ns PC=000c IR=1016, SW=8000, A=    6
 550ns PC=000e IR=3014, SW=8000, A=    6
 570ns PC=0010 IR=2000, SW=8000, A=    6
 590ns PC=0000 IR=0016, SW=8000, A=    3
 610ns PC=0002 IR=401a, SW=8000, A=    3
 630ns PC=0004 IR=5012, SW=8000, A=    3
 650ns PC=0006 IR=1018, SW=8000, A=    4
 670ns PC=0008 IR=3016, SW=8000, A=    4
 690ns PC=000a IR=0014, SW=8000, A=    6
 710ns PC=000c IR=1016, SW=8000, A=   10
 730ns PC=000e IR=3014, SW=8000, A=   10
 750ns PC=0010 IR=2000, SW=8000, A=   10
 770ns PC=0000 IR=0016, SW=8000, A=    4
 790ns PC=0002 IR=401a, SW=8000, A=    4
 810ns PC=0004 IR=5012, SW=8000, A=    4
 830ns PC=0006 IR=1018, SW=8000, A=    5
 850ns PC=0008 IR=3016, SW=8000, A=    5
 870ns PC=000a IR=0014, SW=8000, A=   10
 890ns PC=000c IR=1016, SW=8000, A=   15
 910ns PC=000e IR=3014, SW=8000, A=   15
 930ns PC=0010 IR=2000, SW=8000, A=   15
 950ns PC=0000 IR=0016, SW=8000, A=    5
 970ns PC=0002 IR=401a, SW=8000, A=    5
 990ns PC=0004 IR=5012, SW=8000, A=    5
1010ns PC=0006 IR=1018, SW=8000, A=    6
```

```
1030ns PC=0008 IR=3016, SW=8000, A=    6
1050ns PC=000a IR=0014, SW=8000, A=   15
1070ns PC=000c IR=1016, SW=8000, A=   21
1090ns PC=000e IR=3014, SW=8000, A=   21
1110ns PC=0010 IR=2000, SW=8000, A=   21
1130ns PC=0000 IR=0016, SW=8000, A=    6
1150ns PC=0002 IR=401a, SW=8000, A=    6
1170ns PC=0004 IR=5012, SW=8000, A=    6
1190ns PC=0006 IR=1018, SW=8000, A=    7
1210ns PC=0008 IR=3016, SW=8000, A=    7
1230ns PC=000a IR=0014, SW=8000, A=   21
1250ns PC=000c IR=1016, SW=8000, A=   28
1270ns PC=000e IR=3014, SW=8000, A=   28
1290ns PC=0010 IR=2000, SW=8000, A=   28
1310ns PC=0000 IR=0016, SW=8000, A=    7
1330ns PC=0002 IR=401a, SW=8000, A=    7
1350ns PC=0004 IR=5012, SW=8000, A=    7
1370ns PC=0006 IR=1018, SW=8000, A=    8
1390ns PC=0008 IR=3016, SW=8000, A=    8
1410ns PC=000a IR=0014, SW=8000, A=   28
1430ns PC=000c IR=1016, SW=8000, A=   36
1450ns PC=000e IR=3014, SW=8000, A=   36
1470ns PC=0010 IR=2000, SW=8000, A=   36
1490ns PC=0000 IR=0016, SW=8000, A=    8
1510ns PC=0002 IR=401a, SW=8000, A=    8
```

```
1530ns PC=0004 IR=5012, SW=8000, A=    8
1550ns PC=0006 IR=1018, SW=8000, A=    9
1570ns PC=0008 IR=3016, SW=8000, A=    9
1590ns PC=000a IR=0014, SW=8000, A=   36
1610ns PC=000c IR=1016, SW=8000, A=   45
1630ns PC=000e IR=3014, SW=8000, A=   45
1650ns PC=0010 IR=2000, SW=8000, A=   45
1670ns PC=0000 IR=0016, SW=8000, A=    9
1690ns PC=0002 IR=401a, SW=8000, A=    9
1710ns PC=0004 IR=5012, SW=8000, A=    9
1730ns PC=0006 IR=1018, SW=8000, A=   10
1750ns PC=0008 IR=3016, SW=8000, A=   10
1770ns PC=000a IR=0014, SW=8000, A=   45
1790ns PC=000c IR=1016, SW=8000, A=   55
1810ns PC=000e IR=3014, SW=8000, A=   55
1830ns PC=0010 IR=2000, SW=8000, A=   55
1850ns PC=0000 IR=0016, SW=8000, A=   10
1870ns PC=0002 IR=401a, SW=4000, A=   10
1890ns PC=0004 IR=5012, SW=4000, A=   10
1910ns PC=0012 IR=2012, SW=4000, A=   10
1930ns PC=0012 IR=2012, SW=4000, A=   10
1950ns PC=0012 IR=2012, SW=4000, A=   10
1970ns PC=0012 IR=2012, SW=4000, A=   10
1990ns PC=0012 IR=2012, SW=4000, A=   10
```

您可以看到上述的測試結果中，1+2+...+10 的結果 55 正確的被計算出來

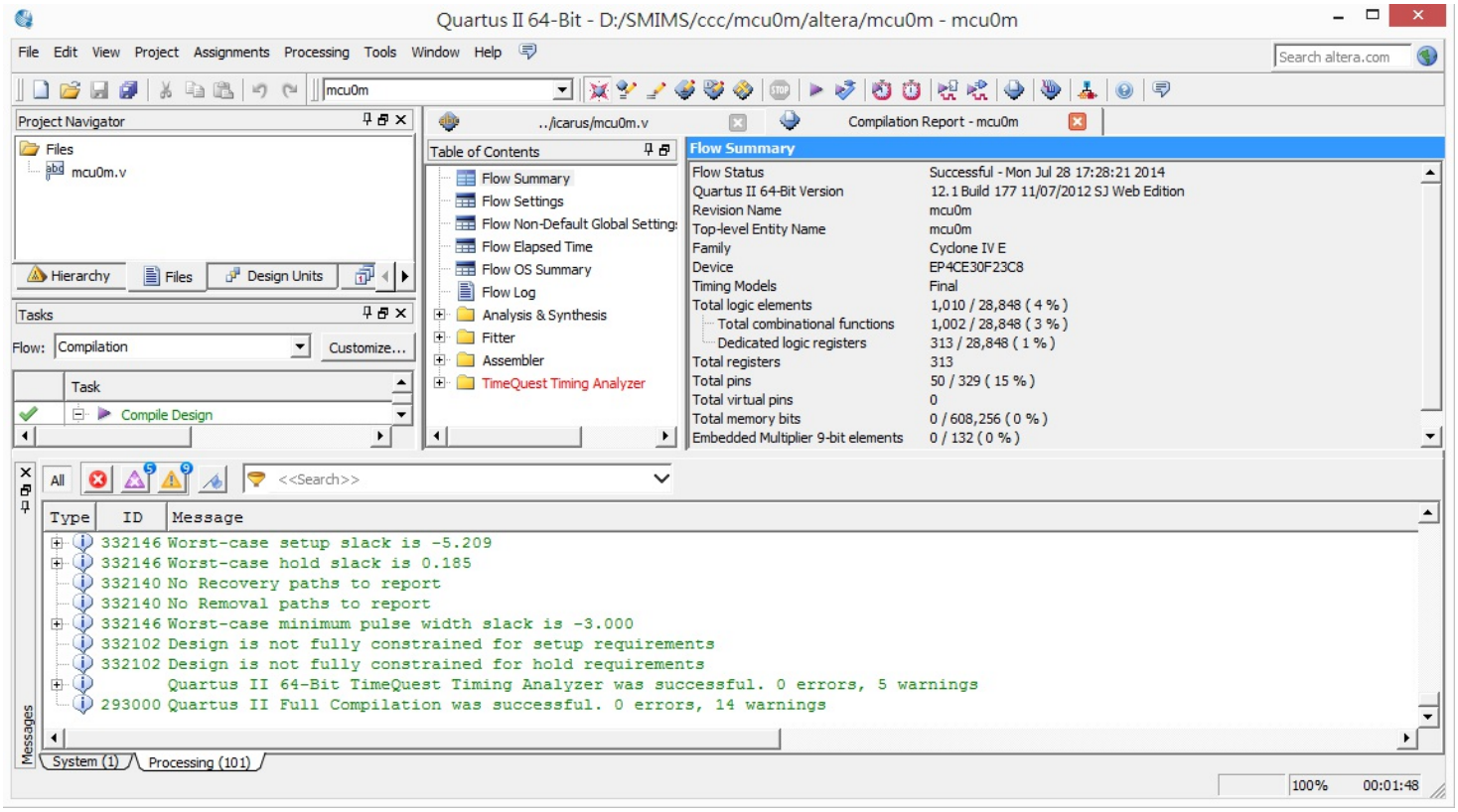
了。

放上 FPGA 實際執行

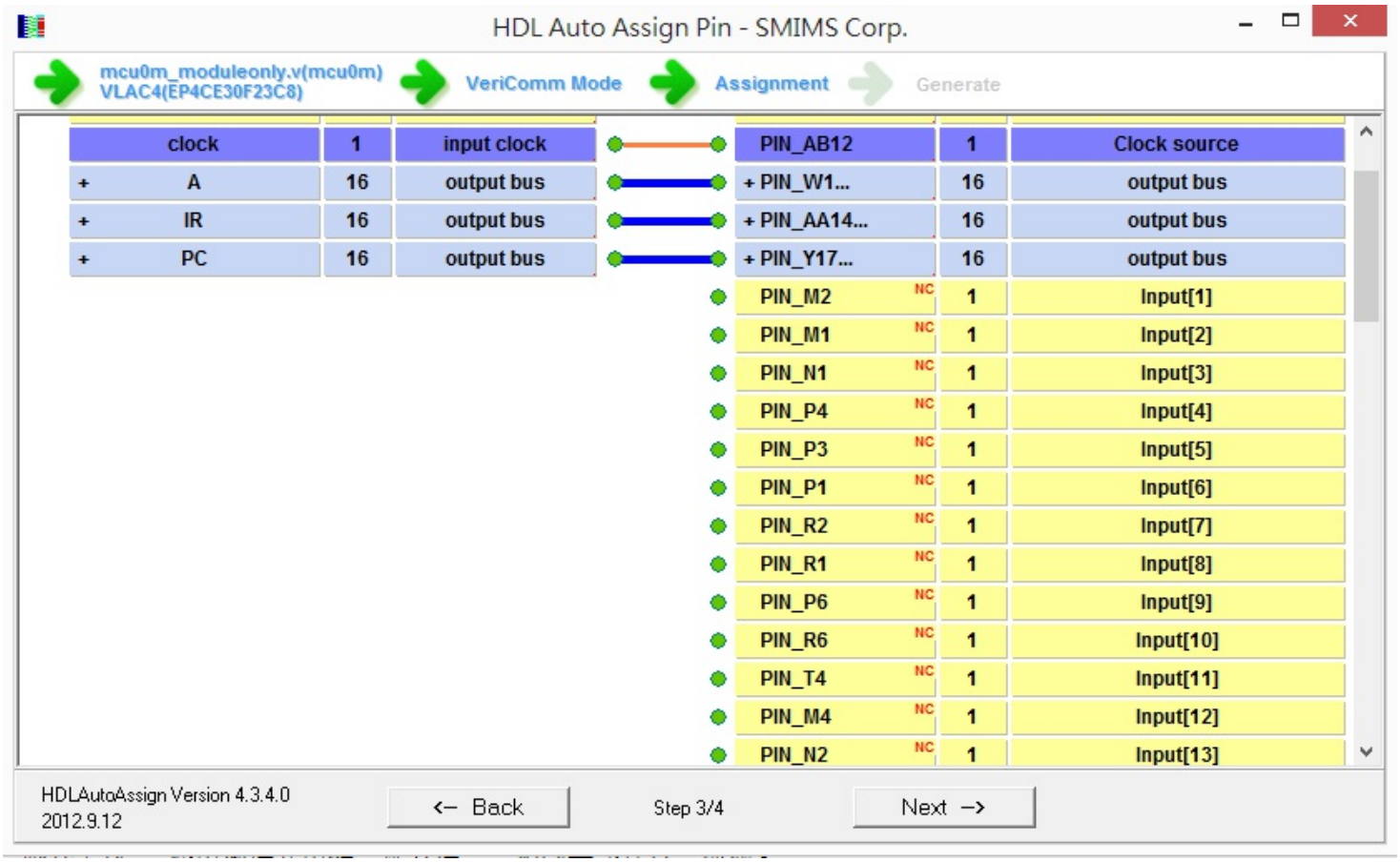
接著、就讓我們將程式放到北瀚科技 (SMIMS) 的「VeriLite Altera C4」FPGA 板上去執行，以下是該 FPGA 板的樣子。

圖、北瀚科技 (SMIMS) 的 VeriLite Altera C4 FPGA 板

由於北瀚的開發工具必須搭配 Altera 的 Quartus II 使用，所以我們必須先將 mcu0m.v 先用 Quartus II 建立專案並且編譯過之後，才能用北瀚的 VeriComm 工具進行燒錄與執行監控等任務，以下是我們用 Quartus II 編譯時的畫面。

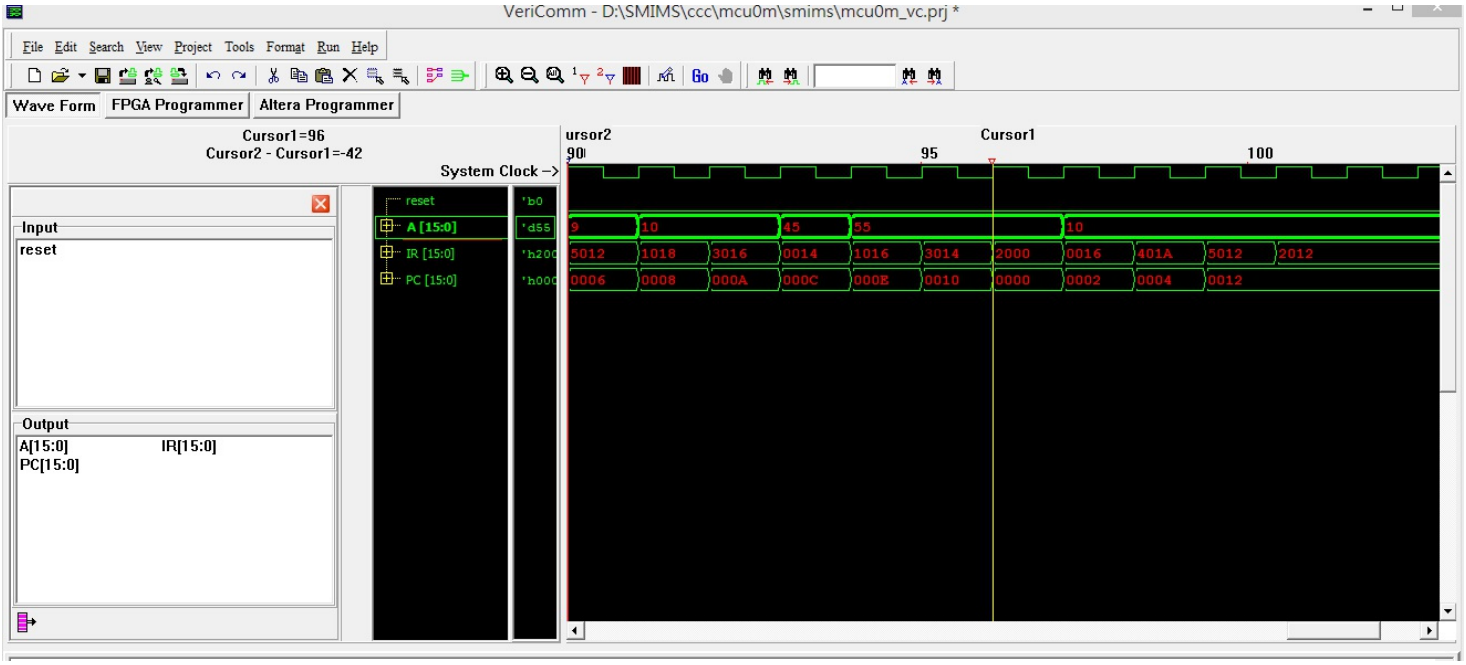


接著我們用北瀚的 HDL Auto Assign Pin 工具產生腳位綁定檔，以下是腳位綁定時的一個畫面。



然後我們將產生的 mcu0m.qsf 檔案複製到 Quartus II 專案中，重新編譯過即可產生燒錄檔 mcu0m.rbf 檔。

接著開啟 VeriComm 並進行燒錄，可以看到該 FPGA 在 VeriComm 監控模式下輸出的波形圖如下。



您可以看到該畫面中正確的計算出 1+2+...+10 = 55 的結果，這代表我們的 mcu0m 模組確實可以放上 FPGA 並且正確的運作了。

結語

必須注意的是，北瀚的 HDL Auto Assign Pin 由於是他們自己寫的一個簡易 Verilog 剖析器，並沒有支援完整的 Verilog 語法，因此在剖析 mcu0m.v 時有失敗的情況，因此筆者將 mcu0m.v 的內容全部刪除之後才交給 HDL Auto Assign Pin 剖析，如此就能正確產生腳位對應檔。

(或許北瀚應該考慮改用 icarus 的剖析器，這樣應該就不用自己辛苦的寫剖析器，也可以支援完整的語法了)

上述編譯、燒錄與執行過程還算蠻漫長的，所以筆者將自己的操作過程錄影了下來，如果本文描述還有不清楚的地方，可以進一步參考下列影片。

- YouTube 影片：開放電腦計畫 - 將 MCU0m 放上北瀚的 FPGA 板執行

從 Arduino 到 AVR 晶片(3) -- Timers（作者：Cooper Maa）

Timers 簡介

簡單地說，Timer 是計時器，可以用來量時間。

來自石英振盪器脈衝 (pulse) 每一個 clock 會來一次，Timer 的內容會跟著計數遞增。所以，如果使用的是 16 MHz 的振盪器，Timer 的內容會每 62.5 ns (奈秒) 改變一次。

根據解析度的不同，Timer 通常有 8-bit 和 16-bit 兩種。如果是 8-bit Timer，那麼可以寫入的最大數值是 255 (16-bit 的話是 65535)，假如超過了最大數值，Timer 就會自動 reset 為 0，這種情況稱為溢位 (overflow)。Timer overflow 的時候可以引發中斷，如果啟用了 Timer overflow 中斷，那麼你就必須在程式裏提供 ISR 處理中斷。

Timer 也可以當成一般的計數器 (Counter) 使用，或者是做 PWM 訊號輸出以及捕捉外部脈衝寬度 (Input Capture)。

Prescaler

Prescaler (預除器) 是一個用來提供 clock 給 Timer 的電路。如你所知，CPU clock 頻率通常是 1 MHz, 8 MHz, 16 MHz，而 Precaler 的用途則是除頻。

AVR 晶片的 Precaler 大部份都有提供底下這些選項:

- No Clock Source (停止 Timer)
- No Prescaling (clock = CPU clock)
- CPU clock / 8
- CPU clock / 64
- CPU clock /256
- CPU clock /1024

Timer 也可以使用外部的 clock，這樣的話，Timer 就變成 Counter (計數器)了。

Timer Registers

ATmega328 有 3 個 Timer/Counters:

- 1 個 8-bit Timer/Counters: Timer0 和 Timer2
- 1 個 16-bit Timer/Counter: Timer1

底下以 Timer0 為例簡介 Timer 的幾個主要暫存器。

首先是 TCCRxA – Timer/Counter Control Register A (x 代表 0, 1 或 2):

TCCR0A – Timer/Counter Control Register A

Bit	7	6	5	4	3	2	1	0	
0x24 (0x44)	COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00	TCCR0A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

TCCRxA 暫存器主要是用來設定 Timer 的模式，例如 PWM 輸出等進階的功能。一般來說，如果沒用到 PWM，只是要單純的 Timer/Counter 功能的話，那麼把 TCCR0A 暫存器設定成 0x00 就行了。

接著是 TCCRxB – Timer/Counter Control Register B:

TCCR0B – Timer/Counter Control Register B

Bit	7	6	5	4	3	2	1	0	
0x25 (0x45)	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00	TCCR0B
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

TCCR0B 暫存器主要是用來設定 clock source。比較重要的是 CS02 CS01 和 CS00 這三個位元，這三個元位就是用來選擇 clock 的:

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/(No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

▲ 註:Timer2 比較特別，有不一樣的 Prescale 設定，請參考 Datasheet

再來是 TCNTx – Timer/Counter Register:

TCNT0 – Timer/Counter Register

Bit	7	6	5	4	3	2	1	0	
0x26 (0x46)	TCNT0[7:0]								TCNT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

TCNTx 暫存器比較簡單，它就是 Timer 的計數器。

最後是 TIMSKx – Timer/Counter Interrupt Mask Register:

TIMSK0 – Timer/Counter Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
(0x6E)	–	–	–	–	–	OCIE0B	OCIE0A	TOIE0	TIMSK0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

TIMSKx 暫存器主要是用來啟用或停用 Timer0 的中斷，透過 TOIE0 這個位元 (Timer/Counter0 Overflow Interrupt Enable)。假如是 Timer1 的話，便是 TIMSK1 暫存器的 TOIE1 這個位元，Timer2 的話是 TIMSK2 的 TOIE2 位元。

【本文作者為馬萬圳，原文網址為：http://coopermaa2nd.blogspot.tw/2011/07/4-timers.html ，由陳鍾誠編輯後納入本雜誌】

泰勒级数（Taylor series）（作者：Bridan）

許多代數式都可以表示為

$$f(x)=\sum_{n=0}^{\infty}\frac{f^{(n)}(a)}{n!}(x-a)^n$$

這稱之為 泰勒級數 展開，如果 a = 0，則改稱為馬克勞林級數展開 (Maclaurin series)

$$f(x)=\sum_{n=0}^{\infty}\frac{f^{(n)}(0)}{n!}x^n$$

如果缺少了這些級數展開，許多電腦數學函數就會計算不出來，在對數一文有一計算式

$$\ln\frac{1+x}{1-x}=2x(1+\frac{x^2}{3}+\frac{x^4}{5}+\cdots)$$

其實它就是一種級數展開，如果電腦要計算三角函數或自然指數，就需要下列算式求值

$$\sin(x)=x(1-\frac{x^2}{3!}+\frac{x^4}{5!}-\frac{x^6}{7!}+\frac{x^8}{9!}-\cdots)\ \ ;for\ all\ x$$

$$\cos(x)=1-\frac{x^2}{2!}+\frac{x^4}{4!}-\frac{x^6}{6!}+\frac{x^8}{8!}-\cdots\ \ ;for\ all\ x$$

$$\tan(x)=x(1+\frac{x^2}{3}+\frac{2x^4}{15}+\frac{17x^6}{315}+\frac{62x^8}{2835}+\cdots)\ \ ;for\ all|x|<\frac{\pi}{2}$$

$$e^x=1+x+\frac{x^2}{2!}+\frac{x^3}{3!}+\frac{x^4}{4!}+\cdots\ \ ;forall\ x$$

如果想計算任意值 u 求其任意次方 y，可令

$$e^x=u^y$$

$$x=y\ln(u)$$

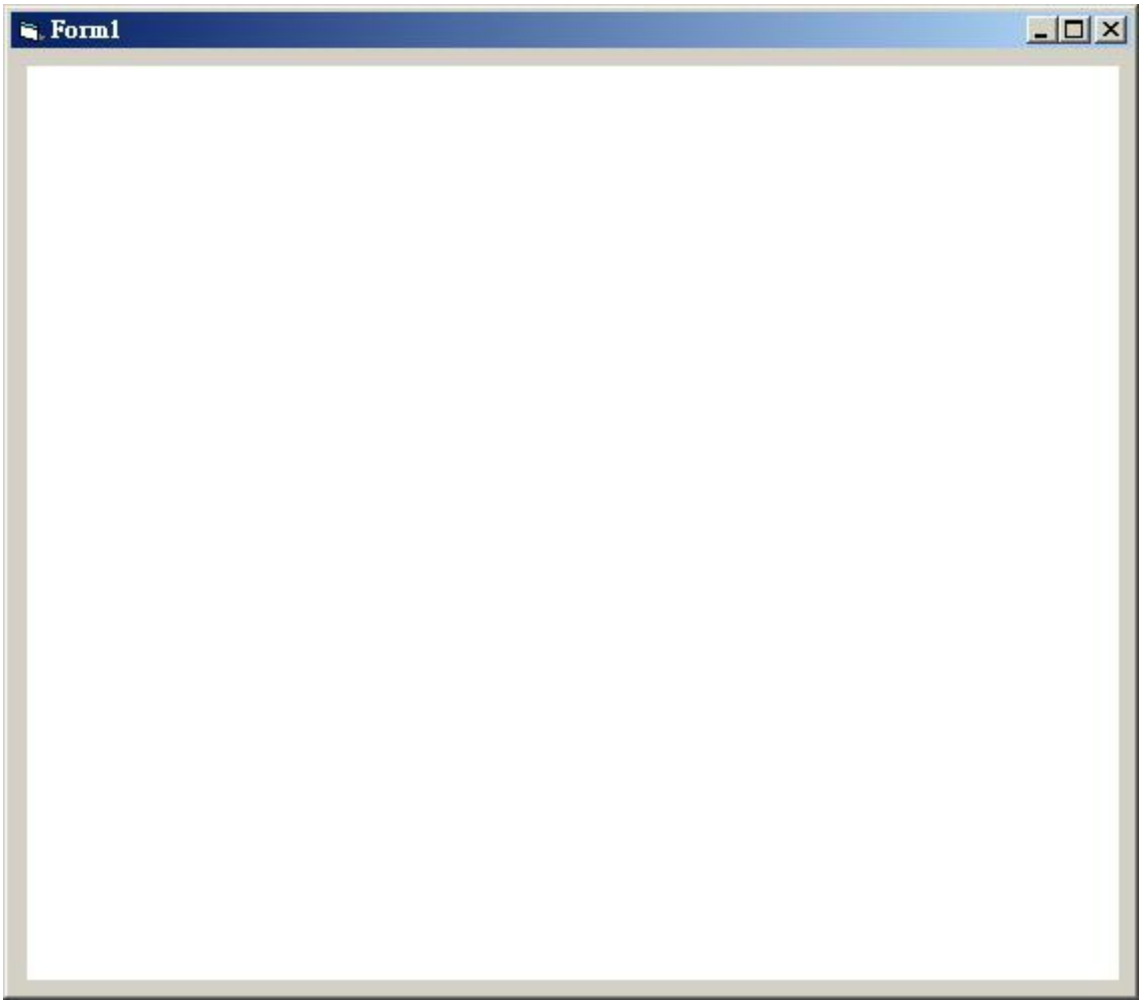
再求解級數和

(本文來自「研發養成所」 Bridan 的網誌，原文網址為 http://4rdp.blogspot.tw/2014/07/taylor-series.html ，由陳鍾誠編輯後納入程式人雜誌)

[Visual Basic 6.0] 利用 WebBrowser 寫 Html 網頁預覽器（作者：廖憲得 0xde）

首先該如何叫出 WebBrowser 瀏覽器元件呢?

專案 => 設定使用元件 找到 "Microsoft Internet Controls" 打勾 => 確定



簡單的 WebBrowser 基礎程式碼

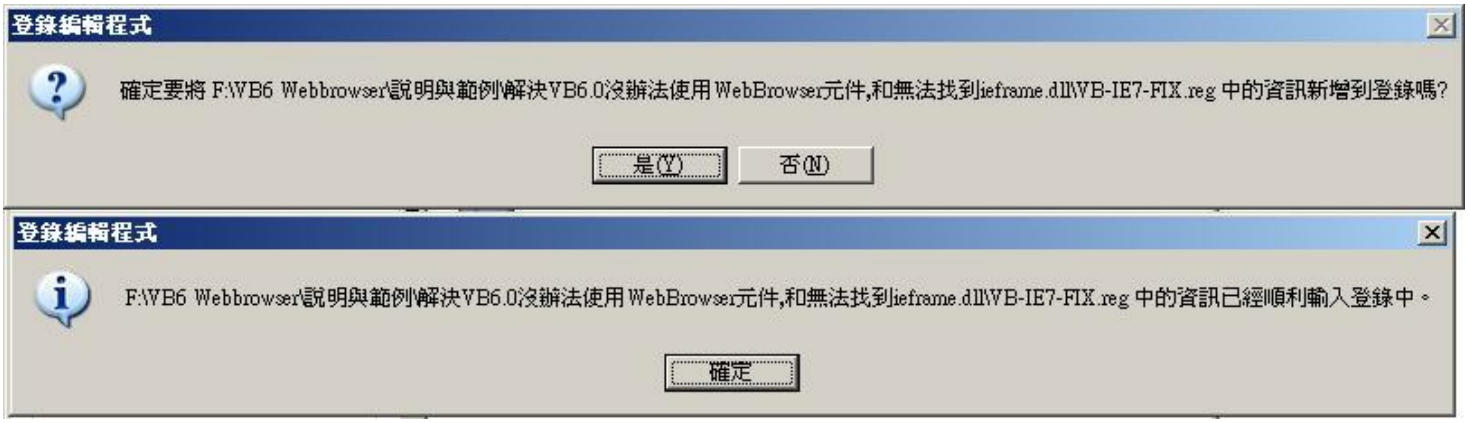
```
上一頁 WebBrowser1.GoBack
下一頁 WebBrowser1.GoForward
瀏覽網址 WebBrowser1.Navigate ("網址")
重新整理 WebBrowser1.Refresh
搜索 WebBrowser1.GoSearch
停止 WebBrowser1.Stop
```

※當出現錯誤訊息：(找不到檔: 'C:32.dll')

將以下 Code 用記事本填入，並另存新檔為 XXX.reg (登錄檔)

```
Windows Registry Editor Version 5.00

[HKEY_CLASSES_ROOT\TypeLib\{EAB22AC0-30C1-11CF-A7EB-0000C05BAE0B}]

[HKEY_CLASSES_ROOT\TypeLib\{EAB22AC0-30C1-11CF-A7EB-0000C05BAE0B}\1.1]
@="Microsoft Internet Controls"

[HKEY_CLASSES_ROOT\TypeLib\{EAB22AC0-30C1-11CF-A7EB-0000C05BAE0B}\1.1\0]

[HKEY_CLASSES_ROOT\TypeLib\{EAB22AC0-30C1-11CF-A7EB-0000C05BAE0B}\1.1\0\win32]
@="C:\\WINDOWS\\system32\\ieframe.dll"
```


解決VB6.0沒辦法使用WebBrowser元件,和無法找到ieframe.dll
檔案(F) 編輯(E) 檢視(V) 我的最愛(A) 工具(T) 說明(H)
上一頁 搜尋 資料夾
網址(D) F:\VB6 Webbrowser\說明與範例\解決VB6.0沒辦法使用WebBrowser元件,和無法找到ieframe.dll
移至
檔案及資料夾工作
將這個檔案重新命名
移動這個檔案
複製這個檔案
將這個檔案發佈到網站
以電子郵件傳送這個檔案
列印這個檔案
刪除這個檔案
其他位置
說明與範例
VB-IE7-FIX
登錄項目
1 KB
類型: 登錄項目 修改日期: 2010/5/30 上午 11:57 大小: 411 個位元組
411 個位元組
我的電腦


登錄編輯程式
確定要將 F:\VB6 Webbrowser\說明與範例\解決VB6.0沒辦法使用WebBrowser元件,和無法找到ieframe.dll\VB-IE7-FIX.reg 中的資訊新增到登錄嗎?
是(Y)
否(N)
登錄編輯程式
F:\VB6 Webbrowser\說明與範例\解決VB6.0沒辦法使用WebBrowser元件,和無法找到ieframe.dll\VB-IE7-FIX.reg 中的資訊已經順利輸入登錄中。
確定

登錄即可。

Html 網頁預覽器

```
Private Sub HtmlTxt_Change()
Open App.Path & "/Html.htm" For Output As #1

Print #1, HtmlTxt

WebBrowserHtml.Navigate App.Path & "/Html.htm"
Close
End Sub
```

1. 在表單上放置一個 WebBrowser 並且將它重新命名為 WebBrowserHtml
2. 在表單上放置一個 TextBox 並且將它重新命名為 HtmlTxt

- 檔案下載：WebBrowser Html 預覽器.rar

【本文作者為「廖憲得」，原文網址為：http://www.dotblogs.com.tw/0xde/archive/2013/11/12/127829.aspx ，由陳鍾誠編輯後納入本雜誌】

雜誌訊息

讀者訂閱

程式人雜誌是一個結合「開放原始碼與公益捐款活動」的雜誌，簡稱「開放公益雜誌」。開放公益雜誌本著「讀書做善事、寫書做公益」的精神，我們非常歡迎程式人認養專欄、或者捐出您的網誌，如果您願意成為本雜誌的專欄作家，請加入 程式人雜誌社團 一同共襄盛舉。

我們透過發行這本雜誌，希望讓大家可以讀到想讀的書，學到想學的技術，同時也讓寫作的朋友的作品能產生良好價值 － 那就是讓讀者根據雜誌的價值捐款給慈善團體。 讀雜誌做公益也不需要有壓力，您不需要每讀一本就急著去捐款，您可以讀了十本再捐，或者使用固定的月捐款方式，當成是雜誌訂閱費，或者是季捐款、一年捐一次等都 OK！甚至是單純當個讀者我們也都很歡迎！

本雜誌每期參考價：NT 50 元，如果您喜歡本雜誌，請將書款捐贈公益團體。例如可捐贈給「羅慧夫顱顏基金會 彰化銀行(009) 帳號：5234-01-41778-800」。(若匯款要加註可用「程式人雜誌」五個字)

投稿須知

給專欄寫作者： 做公益不需要有壓力。如果您願意撰寫專欄，您可以輕鬆的寫，如果當月的稿件出不來，我們會安排其他稿件上場。

給網誌捐贈者： 如果您沒時間寫專欄或投稿，沒關係，只要將您的網誌以 [創作共用的「姓名標示、非商業性、相同方式分享」授權] 並通知我們，我們會自動從中選取需要的文章進行編輯，放入適當的雜誌當中出刊。

給文章投稿者： 程式人雜誌非常歡迎您加入作者的行列，如果您想撰寫任何文章或投稿，請用 markdown 或 LibreOffice 編輯好您的稿件，並於每

個月 25 日前投稿到程式人雜誌社團 的檔案區，我們會盡可能將稿件編入隔月1號出版程式人雜誌當中，也歡迎您到社團中與我們一同討論。

如果您要投稿給程式人雜誌，我們最希望的格式是採用 markdown 的格式撰寫，然後將所有檔按壓縮為 zip 上傳到社團檔案區給我們， 如您想學習 markdown 的撰寫出版方式，可以參考 看影片學 markdown 編輯出版流程 一文。

如果您無法採用 markdown 的方式撰寫，也可以直接給我們您的稿件，像是 MS. Word 的 doc 檔或 LibreOffice 的 odt 檔都可以，我們 會將這些稿件改寫為 markdown 之後編入雜誌當中。

參與編輯

您也可以擔任程式人雜誌的編輯，甚至創造一個全新的公益雜誌，我們誠摯的邀請您加入「開放公益出版」的行列，如果您想擔任編輯或創造新雜誌，也歡迎到 程式人雜誌社團 來與我們討論相關事宜。

公益資訊

公益團體	聯絡資訊	服務對象	捐款帳號
財團法人羅慧夫顱顏基金會	http://www.nncf.org/ lynn@nncf.org 02-27190408分機232	顱顏患者 (如唇顎裂、小耳症或其他罕見顱顏缺陷）	銀行：009彰化銀行民生分行 帳號：5234-01-41778-800
社團法人台灣省兒童少年成長協會	http://www.cyga.org/ cyga99@gmail.com 04-23058005	單親、隔代教養.弱勢及一般家庭之兒童青少年	銀行：新光銀行 戶名：台灣省兒童少年成長協會 帳號：103-0912-10-000212-0